A single model to generate Sanskrit vocabulary.

Bhaskar Kompella

5196, Lovering Drive, Doylestown, PA 18902 USA

email: bhkompella@gmail.com

Abstract

One of the fundamental difficulties in learning संस्कृतम् is to deal with the complexity of the language structure. For example, the need to memorize the very many forms of सुबन्त 's and तिङन्त's baffles learners. Learners must convert their brains into big databases (with a reasonably quick data-retrieval capability) of these forms to become proficient in understanding prose or poetry. The traditional methods depend on the memorization of most of the common forms. This herculean task quickly demotivates learners from learning as they do not have an iota of awareness as to why they should memorize. In this article, we propose a method that starts with providing a big picture of the structure of the language. It suggests systematic structural thinking and lets the learner understand the design of the language in simple terms. We accomplish it in four steps. Firstly, we provide a model structure that acts as the underpinning to all the संस्कृतम् constructs. We call this model structure the “factory”. This structure encapsulates all the mechanisms of word-building (पदनिर्माणम्) in संस्कृतम्. Secondly, we provide the list of factories required for the word-building in संस्कृतम् and the descriptions of the components associated with each factory. Thirdly, we give the interconnections of all the

factories and the mechanisms associated. And, finally, we provide a big picture of the total structure of the language. We would like to reassure readers that this method will not reduce any rules or basic principles of word-building. The recommendation here is that if a संस्कृतम् -learner starts understanding this method and gets a picture of the language structure before diving into learning, he/she will appreciate the logic in शब्दमञ्जरी and धातुमञ्जरी, and कृदन्त, तद्धितान्त constructs, etc. Moreover, the realization that all these constructs follow the same underlying model called ‘factory’, puts him/her at ease and eliminates unnecessary anxiety, and provides some alternative ways of learning. This method, well-known to scientists and engineers, is applied to the linguistic domain in this article. The author is confident that this method can be applied to any language. However, it makes more sense to apply to संस्कृतम्, a language that appears to be overly complex to a new learner.

Introduction

The words (पदम्’s) of a sentence in संस्कृतम् can be broadly divided into three sets: सुबन्त’s, तिङन्त’s, and अव्ययम्’s. A majority of these are not available in a dictionary in the same form as they appear in a sentence. But for some basic अव्ययम्’s, all पदम्’s used in sentences are derived from शब्द ’s and धातु ’s. These derived forms are called grammatical inflections (or inflections). Any dictionary of संस्कृतम् provides basic शब्द ‘s, basic धातु ‘s, basic अव्ययम् ’s. As a result, one must know the mechanisms or methods to build inflections. Learners very frequently (99% of the time) encounter inflections while reading a sentence. Moreover, there exists a big treasure of methodologies that allow learners to build new vocabulary (‘new शब्द ’s, new धातु ’s and new अव्ययम् ’s) from the basic शब्द ’s and धातु ’s. These possibilities of building inflections and new vocabulary stretch the imagination of learners and make them feel lost quickly. Conversely, if a learner encounters an inflection or a derived अव्ययम् (a non-basic अव्ययम्), he/she needs to figure out how it was constructed to get its intended meaning. Every attempt to learn संस्कृतम् will have a store of surprises for the learner.

Learners may wonder if there are rhymes and rules in this practice? Let us formulate the problem at hand as two questions.

1. What are the mechanisms through which the inflections and new vocabulary can be built? (Let us call it ‘the rhyme’ or “WHAT”-part)
2. How do we build them? In other words, what are the rules, methodologies to build these words? (Let us call it ‘the rules’ or “HOW”-part).

To state simply, the first question is asking about the big picture in building the inflections and new vocabulary and, the second question is asking about the detailed ways in which they are built.

In this paper, we attempt to give the big picture that assures the learners that संस्कृतम् has a neat mechanism with which people logically derive the inflections and new vocabulary. It is a simple and elegant mechanism that brings a rhyme into the structure of संस्कृतम्. This rhyme is pivotal to the existence of संस्कृतम्. When teaching संस्कृतम्, every teacher repeats this rhyme multiple times. We believe that highlighting this rhyme upfront and giving a big picture to the learners benefits the learner greatly. We caution the readers that there are some grammatical constructs in संस्कृतम् that will go beyond this model.

In the age of computerization of languages, we believe that this model approach is useful in the efforts to algorithmize the language construction mechanisms.

Preliminaries

In this section, we introduce some definitions and state some well-known facts about संस्कृतम् vocabulary which will be useful in understanding the factory method that we are going to introduce.

In संस्कृतम्, there are some non-words called प्रत्यय 's(suffixes) and उपसर्ग 's (prefixes).

Suffixes are added to शब्द 's and धातु 's to derive inflections of the शब्द 's and धातु 's.

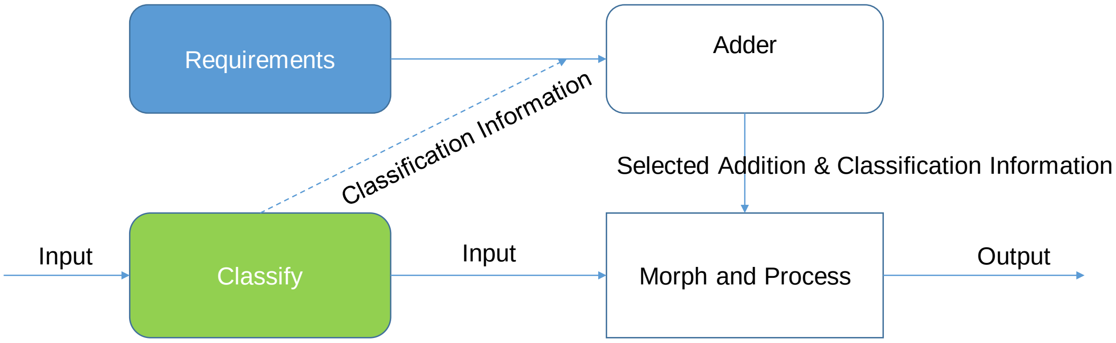
Inflections are of three types. 1) The inflected forms that are अव्यय 's. 2) The inflected forms that create the seven विभक्ति 's (cases) of a शब्द, called सुबन्त 's. 3) The inflected forms that create various verbal forms called तिङन्त 's. Inflections are पदम् 's used in any sentence.

Non-words cannot effect अव्ययम् ‘s. They leave अव्ययम् ‘s unchanged.

One more important fact is that, in संस्कृतम् , new vocabulary can be produced/built from basic धातु ‘s and some basic शब्द ‘s using suffixes and prefixes. In other words, there are established procedures that produce new शब्द ‘s, धातु ‘s, अव्ययम् ‘s. We call production units of the inflections and new vocabulary ‘Factories’. All these factories share a common floor plan. We are going to describe its generic form in the next section.

The Factory Model

Now, we introduce a very generic model called Factory. (see the figure below). As we learn more about this model, we will elaborate on each module of the model in detail. This model can be used to derive the inflections and generate new vocabulary.


Requirements
Adder
Classification Information
Selected Addition & Classification Information
Input
Classify
Input
Morph and Process
Output

Factory – A Generic Model

A factory has the following modules that interact with each other in a prescribed way.

Inputs: These are the raw materials that will be modified to generate outputs that can be utilized for the required purpose.

Classify: This module analyzes the input passed to it and, gets all the classification information of the input and, passes the information to the Adder module.

Requirements: This module specifies the required properties of the anticipating output. These requirements are passed on to the Adder module.

Adder: This module is a decision module. Based on the input classification information and the requirements for the output, this module picks up the right ingredients to produce the final outputs. Also, this module passes the information and the ingredients to the Morph and Process module.

Morph and Process: This module is an operations module. The functions of this module are 1) make any changes that have to be made on the inputs based on the classification information and requirements, 2) make any changes that have to be made to the ingredients based on the classification information and requirements, 3) perform all the operations required to convert inputs into outputs by combining ingredients.

The first and the second functions together are categorized as 'morph function' and the third function is categorized as 'process function'. Each of these functions will have many sequential steps. (When we apply this model to संस्कृतम्, we see that in each function there is a chain of activities grammatically imposed on the input and ingredients. The process function does most of the heavy-duty computation such as applying sandhi rules etc. to finally obtain the output.)

Output: This is the final product matching all the requirements.

This generic framework, the factory model, can be used to derive any inflection and generate new vocabulary. In the following sections, we show how this mechanism can be put into use. We show two major applications of this model and the remaining applications can be easily understood by the readers. These two examples can give the gist, spirit of the model and its applicability in the other areas of संस्कृतम्.

In the following text, we use the term "**run of the factory**" by which we mean that a full cycle (one pass) of sending an input that passes through all the modules and delivering an output.

The सुबन्त Factory – A पदम् generator

In this section, we apply the factory model in deriving सुबन्त 's. We describe each module and the jobs they do in deriving सुबन्त 's. We omit the grammatic details and only capture the events relevant to the application of the model. We use only one example to explain the application. Readers can apply this model to any शब्द of any gender and with any ending.

The input of the सुबन्त factory**:** शब्द 's are the inputs of this factory. We need to pass only one शब्द as input for each run of the factory. For example राम.

Classify module of the सुबन्त factory: In this module, the शब्द will be classified. In the given example, this module inspects and determines that राम is अकारान्तः (ending with अ) and पुंलिङ्गशब्दः (masculine gender). This classification information will be passed to the Adder module.

Requirements module of the the सुबन्त factory: We need to specify two requirements in each pass. Firstly, the विभक्ति (case) of the शब्द we are looking for and, secondly, वचनम् (the number) of the output we are looking for. For example, we can specify the requirements as 1) सप्तमी विभक्ति (Genitive Case) of राम and 2) बहुवचनम् (plural number).

Adder module of the सुबन्त factory: In this factory, there are 21 suffixes (called सुप् 's). In each run of the factory, an appropriate suffix will be picked based on the requirements from the requirement module. In our example, Adder picks आम् from सुप्'s as per the requirements. (It is easy to make the suffix count. 7 cases x 3 numbers = 21 सुप् 's)

Morph and Process module of the सुबन्त factory: To start with, this module uses some standard grammatical mechanisms to morph the input (whenever necessary) considering the classification of the input. In many cases, the morph function leaves the input unaltered in this factory. The result of the morph function is passed to the process function. The process function राम + आम् (*) of applying appropriate grammatical rules to finally obtain the output. In our example, there is no morphing required. When the input passed to the process function produces रामाणाम् as the result due to the classification properties of the input.

(*) For the readers interested in knowing all operations, we provide the Panini rules applied below.

राम + आम् (स्वौजसमौट्छष्टाभ्याम्भिस्ङेभ्याम्भ्यस्ङसिभ्याम्भ्यस्ङसोसाम्ङ्योस्सुप् ४.१.२)

→ राम + नुट् + आम् (ह्रस्वनद्यापो नुट् ७.१.५४)

→ राम + न् + आम् (हलन्त्यम् १.३.३, तस्य लोपः १.३.९)

→ रामानाम् (नामि ६.४.३)

→ रामाणाम् (अट्कुप्वाङ्नुम्व्यवायेऽपि ८.४.२).

The output of the सुबन्त factory: The required inflection is obtained at this stage. In our example, रामाणाम् is the plural genitive inflection of रामशब्दः

Remark 1: सुप् contains 21 suffixes and picking up the right suffix from सुप् according to the requirement is the easiest exercise. But, depending on the classification properties of the input, the conjugated form of input and the suffix may result in a quite different sounding inflection. This is one of the major complexities of संस्कृतम् which can not be avoided. For most of the input classifications (for example, अजन्त's, हलन्त's, etc. in all three genders), the output has a consistent pattern. In other words, the morph and process functions give consistent results. For example, नर is अकारान्तः and पुंलिङ्गः. Its genetic plural form is नराणाम् . Similarly, शिव is another अकारान्तः and पुंलिङ्गः for which the genetic plural form is शिवानाम् . However, there are some exceptions which were listed in शब्दमन्जरी in the category of विशेषशब्दाः

The तिङन्त Factory – A पदम् generator

Now, let us use the factory method to derive तिङन्त 's. We briefly mention a run of the factory in this case.

The input of the तिङन्त factory: धातु 's are the inputs of this factory. We need to pass only one धातु as input for each run of the factory. For example, चरँ.

Classify module of the तिङन्त factory: In this module, the धातु will be classified. In the given example, this module inspects and classifies चरँ as भ्वादिगण, सकर्मकम् , परस्मैपदी and सेट्. This classification information will be passed to the Adder module.

Requirements module of the तिङन्त factory**:** We need to specify three requirements in each pass. 1) the tense or mood of the धातु we are looking for, 2) पुरुष: (the person) of the output and 3) वचनम् (the number) of the output. For example, we can specify the requirements as वर्तमानकालः (लट्) (present tense), उत्तमपुरुषः (first person), एकवचनम् (singular).

Adder module of the तिङन्त factory**:** In this factory, there are 180 suffixes (called तिङ ‘s). The adder module utilizes the classification information to pick the right suffix from these 180 suffixes. In the case of our example, it picks मिप्. (It is easy to make the suffix count. 10 tenses and moods x 3 numbers x 3 genders = 90. 90 for परस्मैपदी + 90 for आत्मनेपदी = 180 तिङ ‘s).

Morph and Process module of the तिङन्त factory**:** The morph function converts चरँ to चर् taking the classification properties and requirement. Then the process function conjugates using चर् with मिप् by using appropriate grammatical rules** to give the result चरामि. It is important to note that both morph and process functions play a very important role in this factory.

(**) For the readers interested in knowing all operations, we provide the Panini rules applied below.

चरँ

→ चरँ (भूवादयो धातवः१.३.१]

→ चरँ [उपदेशेऽजनुनासिक इत्१.३.२]

→ चर् [तस्य लोपः१.३.९]

→ चर् [सनाद्यन्ता धातवः३.१.३२]

→ चर् [लः कर्मणि च भावे चाकर्मकेभ्यः।३.४.६९]

→ चर्+लँट् [वर्तमाने लट्३.२.१२३]

→ चर्+लँट् [लस्य३.४.७७]

→ चर्+लँट् [हलन्त्यम्१.३.३]

→ चर्+लँ [तस्य लोपः१.३.९]

→ चर्+लँ [उपदेशेऽजनुनासिक इत्१.३.२]

→ चर्+ल् [तस्य लोपः१.३.९]

→ चर्+मिप् [तिप्तस्झिसिप्थस्थमिब्वस्मस् तातांझथासाथांध्वमिड्वहिमहिङ्३.४.७८]

→ चर्+मिप् [लः परस्मैपदम्१.४.९९]

→ चर्+मिप् [तिङस्त्रीणि त्रीणि प्रथममध्यमोत्तमाः१.४.१०१]

→ चर्+मिप् [तान्येकवचनद्विवचनबहुवचनान्येकशः१.४.१०२]

→ चर्+मिप् [अस्मद्युत्तमः१.४.१०७]

→ चर्+मिप् [तिङ्शित्सार्वधातुकम्३.४.११३]

→ चर्+शप्+मिप् [कर्तरि शप्३.१.६८]

→ चर्+शप्+मिप् [तिङ्शित्सार्वधातुकम्३.४.११३]

→ चर्+शप्+मिप् [लशक्वतद्धिते१.३.८]

→ चर्+अप्+मिप् [तस्य लोपः१.३.९]

→ चर्+अप्+मिप् [हलन्त्यम्१.३.३]

→ चर्+अ+मि [तस्य लोपः१.३.९]

→ चरा+मि [अतो दीर्घो यञि७.३.१०१]

→ चरामि

The output of the तिङन्त factory: The required reflection is obtained at this stage. In our example, चरामि is the present tense inflection of चर in the first person singular.

Remark 2: From the above two example factories, we can understand that the Morph and Process module is the backbone of the inflection generation. Appropriate grammar rules have to be applied in this module carefully. The knowledge of the grammar of संस्कृतम् (like सिद्धान्तकौमुदि or अष्टाध्यायी) is essential to perform the operations of this modules. In this way, the factory method will preserve the grammatical rules. However, the awareness that all inflections are generated by the above steps in a standardized way (by a factory model) keeps the learners' minds at ease. The inflection generation process looks logical and the tedious exercise of memorizing शब्दमञ्जरी and धातुमञ्जरी sounds meaningful to them. Here is the good news. Thanks to the internet technologies, learners do not need to memorize शब्दमञ्जरी and धातुमञ्जरी. We have the inflected forms at a few clicks on a mouse (https://ashtadhyayi.com/).

Vocabulary Building Factories

The outputs of the above two factories are inflections of शब्द 's and धातु 's. In other words, we generate पद 's with the above factories. The factories we mention in this section have a different flavor. Their outputs are new शब्द ‘s, new धातु ‘s, new अव्ययम् ‘s. One can use these outputs to generate पद ‘s using the above two factories. We mention the inputs and the adders that play role in the factories. All other details can be worked out by readers in the lines of the above examples.

A शब्द factory can be built with the following inputs and suffixes:

Input	Adder Module	Output
शब्दः	Some तद्धित suffixes	शब्दः
धातुः	Some कृत् suffixes	शब्दः
पुरुषवाचकशब्दः	आ, ऊ, ई, ति suffixes	स्त्रीवाचकशब्दः

An अव्ययम् factory can be built with the following inputs and suffixes:

Input	Adder Module	Output
धातुः	Some कृत् suffixes such as क्त्वा, ल्यप्, तुमुन्, णमुल्	अव्ययम्
शब्दः	Some तद्धित suffixes	अव्ययम्

A धातु factory can be built with suffixes and prefixes:

Input	Adder Module	Output
धातुः	णिच्, सन्, यङ्, यङ्लुग् etc. as suffixes	धातुः
धातुः	21 prefixes (उपसर्गाः)	धातुः
धातुः	Some विशेषणशब्द’s as prefixes	धातुः
धातुः	Some अव्ययम्’s as prefixes	धातुः
शब्दः	नामधातु suffixes	धातुः

Conclusions

The factory approach proposed in this paper has the following advantages:

1. The approach separates WHAT TO DO? from HOW TO DO? (Rhyme from Rule).
2. The procedures of WHAT-part are encapsulated in a single model.
3. The Factory model is applicable in deriving inflections and new vocabulary which constitute a big part of the grammatical constructs. (We have not covered some important topics such as समासविधि etc.).
4. The WHAT-parts of most of the complex topics of संस्कृतम् are covered under this model. The model provides a big, unified picture of these grammatical constructs. The awareness of WHAT-part puts learners at ease. They understand that the details can be worked out whenever necessary.
5. The factory model will provide a systematic thinking process for the linguistic scientist to computerize the grammatical constructs. Building a rule engine that performs

required operations for the Morph and Process module is a big challenge. However, one can overcome the challenge with the well-ordered and comprehensive rule sets of Maharshi Panini's अष्टाध्यायी [1]. A great deal of research is in progress in this area.

6. संस्कृतम् can be very scientifically built starting with a basic set of { शब्द 's , धातु 's, suffixes and prefixes } and then applying 3981 sutras of अष्टाध्यायी [1] on this basic set in various ways. The factory model makes readers appreciate this fact very quickly. (ref: the third point in this section.)

---

1. S. C. Vasu, *"The Ashtadhyayi of Panini: Volumes 1 thru 8 + Intro".*